UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/659,695 | 09/10/2003 | Scott A. Abfalter | 94022.8300 | 7542 |

86244 7590 12/06/2010
Snell & Wilmer L.L.P., (Barker)
One Arizona Center
400 East Van Buren Street
Pheonix, AZ 85004-2202

| EXAMINER | |
|---|---|
| WANG, JUE S | |
| **ART UNIT** | **PAPER NUMBER** |
| 2193 | |
| **NOTIFICATION DATE** | **DELIVERY MODE** |
| 12/06/2010 | ELECTRONIC |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

dbarker@swlaw.com
landerson@swlaw.com
ccrawford@swlaw.com

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/659,695 | ABFALTER ET AL. | |
| | Examiner | Art Unit | |
| | JUE WANG | 2193 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

## Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

## Status

1)☒ Responsive to communication(s) filed on _01 October 2010_.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

## Disposition of Claims

4)☒ Claim(s) _1,3,6-17,19-22,25-28,30-36 and 39_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1, 3, 6-17, 19-22, 25-28, 30-36, and 39_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

## Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

## Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

| | |
|---|---|
| 1) ☒ Notice of References Cited (PTO-892) | 4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ . |
| 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) ☐ Notice of Informal Patent Application |
| 3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _11/29/2010_. | 6) ☐ Other: _____. |

## DETAILED ACTION

1. Claims 1, 3, 6-17, 19-22, 25-28, 30-36, and 39 have been examined.

2. Claims 4 and 5 were cancelled in Amendment dated 5/21/2008. Claims 2, 29, 37, and 38 were cancelled in Amendment dated 1/27/2009. Claims 23 and 24 were cancelled in Amendment dated 7/12/2009. Claim 18 was cancelled in Amendment dated 12/11/2009.

### *Claim Rejections - 35 USC § 112*

3. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

4. Claims 1, 3, 6-17, 19-22, 25-28, 30-36, and 39 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

5. Independent claims 1, 16, 27, and 39 recite "automatically switching from said selected software application to a different version of said selected application in response to a first fault, without uninstalling said selected software application; and uninstalling said selected software application and reverting back to said different version of said selected software application in response to a second fault." Applicant state that support for these limitations can be found at paragraphs 37-40 of the specification.

Paragraph 37 of the specification recites "the error detection application 164 can interface with uninstall application to initiate an uninstall of a particular software version of a fault is detected." Paragraph 38 of the specification recites "In yet another arrangement, a version of the software which is loaded on restart can depend on the error which is detected. For example, if a particular error signal is predefined as being associated with a software/hardware incompatibility, a previous version of the software can be loaded on the next restart. … However, if the error is not so predefined, the current version of the software can be re-loaded on the restart. If after a predefined number of attempts, for instance 2, the software cannot be successfully loaded, the restart application can trigger the previous software version to be loaded." From these two passages, Examiner submits that the specification recites selecting the version of software to load (i.e., the current version or a previous version) depending on the error and does not recite selecting whether or not to uninstall the selected software application as recited in the claims. The specification recites that a particular software version can be uninstalled if a fault is detected, however, this appears to be an operation that can be performed when it is determined that the previous version of software is to be loaded and the specification does not suggest reverting to a prior version without uninstalling in response to a first fault and uninstall and reverting to a prior version in response to a second fault.

6. Any claim not specifically addressed, above, is being rejected as incorporating the deficiencies of a claim upon which it depends.

***Claim Rejections - 35 USC § 103***

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1, 3, 6-16, 19-22, 26, and 39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kraml et al. (US 6,141,683, hereinafter Kraml), in view of Aghera et al. (US 2004/0098715 A1, hereinafter Aghera), further in view of Cheston et al. (US 6,195,695 B1, hereinafter Cheston), further in view of Linam et al. (US 6,401,218 B1, hereinafter Linam).

9. As per claim 1, Kraml teaches the invention as claimed, including a method comprising:

transferring, via wireless communication, software directly to a computing device from a software server to create transferred software, said software server remotely located with respect to said computing device, wherein said transferred software is another version of software currently running in said computing device, and wherein said transferred software is stored in to at least a portion of a data store associated with said computing device (i.e., "control center 210 advantageously transmits the newest version of software application, version n+1 to remote computer 203, …, remote computer 203, under the control of version n, stores version n+1", see Fig 2, Fig 3, Fig 4, steps 401, 402, column 4, lines 29-62, column 5, lines 55-56; EN: version n+1 is transferred and it is another version of the software running on the device); and

sending an instruction via wireless communication directly to said computing device identifying said transferred software or said software currently running in said computing device as a selected software application to be loaded by said computing device in response to a restart of said computing device (i.e., “control center 210 transmits a command to remote computer 230 directing remote computer 230 to store into pointer 330 the address of the location in first memory 340 where version n+1 is stored”, see column 6, lines 5-8, “control center 210 transmits a command to remote computer 230 directing remote computer 230 to store the address of the location of version n into pointer 330”, see column 6, lines 58-60; EN: the command to the remote computer is the instruction sent and the command identifies the software to be loaded by instructing the remote computer to store the address of the location of version n or n+1 into the pointer); and

automatically switching from said selected software application to a different version of said selected software application in response to a first fault, without uninstalling said selected software application (i.e., “remote computer 230 determines, if possible, if version n+1 has crashed, …, control center 210 advantageously receives the message indicating that version n+1 has crashed, ... if a roll-back to version n should be initiated”, see Fig 4, steps 411, 412, 415, 416, column 7, lines 10-42).

Kraml does not explicitly teach that the computing device is a software defined radio device and the software and instructions are transferred via radio frequency (RF) communication.

Aghera is cited to teach updating software in a software defined radio device and the software and update instructions are transferred via radio frequency (RF) communication (see abstract, Fig 1, [0002], [0022]-[0024], [0053-[0060]).

It would have been obvious to one of ordinary skill in the art at the time of the invention to use the method of updating software as taught by Kraml in a software defined radio device using radio frequency (RF) communication as described by Aghera because Kraml does not limit the type of computing device on which the method of software updating can be performed (see column 3, lines 20-24, column 4, lines 36-40 of Kraml) and it is advantageous to use the updating method of Kraml in other types of computing devices such as a software defined radio device to benefit from the rollback feature such that if the new version of the software application is or becomes unusable for any reason, the device can quickly roll-back to the older version (see column 3, lines 33-37 of Kraml).

Kraml and Aghera do not explicitly teach transferring, via said RF communication, a back-up copy of said transferred software that is executed in response to runtime errors, wherein said runtime errors are generated by executing said transferred software.

Cheston teaches a method of transferring software to a computer for execution, transferring, via network communication, a backup copy of the transferred software that is executed in response to runtime errors, wherein said runtime errors are generated by executing said transferred software (i.e., obtaining another copy of the application program after the application has become corrupted and crashed, see column 1, lines 34-51).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml and Aghera to transfer, via said RF communication, a back-up copy of said transferred software that is executed in response to runtime errors, wherein said runtime errors are generated by executing said transferred software as similarly taught by Cheston because executable applications often becomes corrupted and crashes during operation and another copy of the application is needed in order recover from the crash (see column 1, lines 38-51 of Cheston).

Kraml, Aghera, and Cheston do not teach uninstalling said selected software application and reverting back to said different version of said selected software application in response to a second fault.

Linam teaches uninstalling an update to a software component in response to an error detected during functional verification testing (see Fig 3, column 1, line 60 – column 2, line 7, column 3, lines 43-57, column 4, lines 10-43).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml, Aghera, and Cheston to uninstall said selected software application and reverting back to said different version of said selected software application in response to a second fault as taught by Linam such that the selected software application can be uninstalled based on functionality verification testing which would indicate software errors (see Fig 3, column 3, lines 43- column 4, line 42 of Linam) instead of waiting for the software to crash to revert to another version as taught by Kraml (see column 7, lines 10-29 of Kraml).

10. As per claim 3, Kraml teaches monitoring said transferring of said transferred software and monitoring said loading of said selected software application (see column 6, lines 29-38).

11. As per claim 6, Kraml teaches wherein said instruction identifies a software version (see Fig 4, steps 403, 412, column 6, lines 5-8, 58-61).

12. As per claim 7, Aghera teaches wherein said software-defined radio device comprises a plurality of software defined radio devices (see [0024]).

13. As per claim 8, Kraml teaches receiving an error indication in response to said first fault or second fault being detected in at least one of said transferring of said transferred software or said loading of said selected software application (see Fig 4, step 409, column 6, lines 52-54).

14. As per claim 9, Kraml does not explicitly teach transferring software that comprises a plurality of software components.

Aghera teaches transferring software that comprises a plurality of software components (see [0032]).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml such that the transferred software comprises a plurality of software components as taught by Aghera because it is well known in the art to structure software as one or more components to conform to the well known practice of abstraction and encapsulation.

15. As per claim 10, Kraml does not explicitly teach receiving a version indicator from said software-defined radio device, said version indicator identifying software which is currently loaded on said software-defined radio device.

Aghera teaches receiving a version indicator from said software-defined radio device, said version indicator identifying software which is currently loaded on said software-defined radio device (see [0026], [0045]).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml to receive a version indicator from said software-defined radio device, said version indicator identifying software which is currently loaded on said software-defined radio device as taught by Aghera such that the patch server application can check if the wireless device needs a software patch (see [0045] of Aghera).

16. As per claim 11, Kraml does not explicitly teach receiving a software listing from said software-defined radio device, said software listing identifying software currently available on said data store.

Aghera teaches receiving a software listing from said software-defined radio device, said software listing identifying software currently available on said data store (i.e., patch profile maintains version information of all upgradeable software component on the wireless device, see [0030]).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml to receive a software listing from said software-defined radio device, said software listing identifying software currently available on said data store as taught by Aghera such that the patch server will be able to determine whether or not the wireless device requires a certain patch (see [0030] of Aghera).

17. As per claim 12, Kraml as modified teaches wherein said transferred software is stored in a second data store associated with said software-defined device (see column 5, lines 55-56).

18. As per claim 13, Kraml teaches wherein said second data store is nonvolatile (see column 5, lines 1-8).

19. As per claim 14, Kraml as modified teaches wherein said transferring of said transferred software occurs in response to said software-defined radio device continuing to perform software-defined radio functions (see column 65, line 65 - column 6, line 4).

20. As per claim 15, Kraml as modified teaches wherein said software server comprises a computer operatively connected to said software-defined radio device via a wireless communications network (see Fig 2, column 4, lines 29-49).

21. As per claim 16, Kraml teaches the invention as claimed, including a method comprising:

receiving, via wireless communication directly from a software server, transferred software at a computing device, said software server remotely located with respect to said computing device, wherein said transferred software is another version of software currently running in said computing device i.e., “control center 210 advantageously transmits the newest version of software application, version n+1 to remote computer 203, …, remote computer 203, under the control of version n, stores version n+1”, see Fig 2, Fig 3, Fig 4, steps 401, 402, column 4, lines 29-62, column 5, lines 55-56; EN: version n+1 is transferred and it is another version of the software running on the device), and wherein said software currently running in said computing device is stored in a first non-volatile data store area (see column 5, lines 1-8, 48-50, column 6, lines 58-61; EN: version n is stored at a location in first memory which is a non-volatile memory);

storing said transferred software in a second non-volatile data store area distinct from said first non-volatile data store area associated with said computing device (see column 5, lines 1-8, 48-50, 55-57, column 6, lines 5-20, 58-61; EN: version n+1 is stored into a different location within the first memory);

receiving, via wireless communication directly from said software server, an instruction at said computing device identifying said transferred software or said software currently running in said computing device as a selected software application to be loaded by said computing device in response to a restart of said computing device (i.e., the command to remote computer to store the address of the location of version n or n+1 into pointer, see Fig 1, Fig 2, Fig 4, steps 401, 403, 405, 412, column 1, lines 40-45, column 5, lines 45-50, column 6, lines 15-61); and providing an error indication in response to a first fault detection, and selecting a different software version of said selected software application based on said a particular error in said error indication, without uninstalling said selected software application (i.e., "remote computer 230 determines, if possible, if version n+1 has crashed, …, control center 210 advantageously receives the message indicating that version n+1 has crashed, ... if a roll-back to version n should be initiated", see Fig 4, steps 411, 412, 415, 416, column 7, lines 10-42);

responsive to a restart instruction, restarting said computing device and loading said selected software application (see Fig 4, steps 406, 407, 414, column 6, lines 25-38, column 7, lines 4-9); and

verifying said selected software application is loaded successfully (see column 6, lines 29-38, column 7, lines 10-11).

Kraml does not explicitly teach that the computing device is a software defined radio device and the software and instructions are transferred via radio frequency (RF) communication.

Aghera is cited to teach updating software in a software defined radio device and the software and update instructions are transferred via radio frequency (RF) communication (see abstract, Fig 1, [0002], [0022]-[0024], [0053-[0060]).

It would have been obvious to one of ordinary skill in the art at the time of the invention to use the method of updating software as taught by Kraml in a software defined radio device using radio frequency (RF) communication as described by Aghera because Kraml does not limit the type of computing device on which the method of software updating can be performed (see column 3, lines 20-24, column 4, lines 36-40 of

Kraml) and it is advantageous to use the updating method of Kraml in other types of computing devices such as a software defined radio device to benefit from the rollback feature such that if the new version of the software application is or becomes unusable for any reason, the device can quickly roll-back to the older version (see column 3, lines 33-37 of Kraml).

Kraml and Aghera do not explicitly teach transferring, via said RF communication, a back-up copy of said transferred software that is executed in response to runtime errors, wherein said runtime errors are generated by executing said transferred software.

Cheston teaches a method of transferring software to a computer for execution, transferring, via network communication, a backup copy of the transferred software that is executed in response to runtime errors, wherein said runtime errors are generated by executing said transferred software (i.e., obtaining another copy of the application program after the application has become corrupted and crashed, see column 1, lines 34-51).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml and Aghera to transfer, via said RF communication, a back-up copy of said transferred software that is executed in response to runtime errors, wherein said runtime errors are generated by executing said transferred software as similarly taught by Cheston because executable applications often becomes corrupted and crashes during operation and another copy of the application is needed in order recover from the crash (see column 1, lines 38-51 of Cheston).

Kraml, Aghera, and Cheston do not teach uninstalling said selected software application and reverting back to said different version of said selected software application in response to a second fault.

Linam teaches uninstalling an update to a software component in response to an error detected during functional verification testing (see Fig 3, column 1, line 60 – column 2, line 7, column 3, lines 43-57, column 4, lines 10-43).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml, Aghera, and Cheston to uninstall said selected software application and reverting back to said different version of said selected software application in response to a second fault as taught by Linam such that the selected software application can be uninstalled based on functionality verification testing which would indicate software errors (see Fig 3, column 3, lines 43- column 4, line 42 of Linam) instead of waiting for the software to crash to revert to another version as taught by Kraml (see column 7, lines 10-29 of Kraml).

22. As per claim 19, Kraml teaches monitoring said receiving transferred software step and providing an error indication in response to a fault being detected in said receiving transferred software step (see column 6, lines 29-46).

23. As per claims 20-22 and 26, these claims recite limitations that are substantially similar to the limitations of claims 6, 10, 11, and 14. Therefore, they are rejected using the same reasons as claims 6, 10, 11, and 14.

24. As per claim 39, the limitations recited in this computer-readable medium claim are substantially similar to the limitations recited in claim 16. Therefore, it is rejected using the same reasons as claim 16.

25. Claims 17, 27, 28, and 31-36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kraml et al. (US 6,141,683, hereinafter Kraml), in view of Aghera et al. (US 2004/0098715 A1, hereinafter Aghera), further in view of Cheston et al. (US 6,195,695 B1, hereinafter Cheston), further in view of Linam et al. (US 6,401,218 B1, hereinafter Linam), further in view of Aija et al. (US 6,928,579 B2, hereinafter Aija).

26. As per claim 17, Kraml as modified teaches automatically reverting from said selected software application to a different software version, wherein said automatically reverting is in response to said selected software application encountering an error which causes said software-defined radio device to stop functioning properly (see column 7, lines 10-43).

Kraml does not explicitly teach that the reverting is performed without an instruction from said software server.

Aija teaches automatically reverting from a version of the software to a different software version without an instruction from a software server (see column 5, lines 51-60).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml to automatically revert to a different software version without an instruction from said software server as taught by Aija such that the client device can initiate the recovery process after a system crash (see column 5, lines 51-53 of Aija).

27. As per claim 27, Kraml teaches the invention as claimed, including a device comprising:

a communication interface configured to receive transferred software and an instruction directly from a software server remotely located with respect to said device (see Fig 3, item 310, column 4, lines 54-57), wherein said transferred software is another version of software configured to be currently running in said device (see column 5, lines 45-54; EN: version n+1 is transferred and it is another version of the software running on the device), and wherein said software server provides an instruction comprising a selected software configured to specify whether said transferred software or said software configured to be currently running in said device will be loaded in response to a restart of said device (i.e., the command to remote computer to store the address of the location of version n or n+1 into pointer, see Fig 1, Fig 2, Fig 4, steps 401, 403, 405, 412, column 1, lines 40-45, column 5, lines 45-50, column 6, lines 15-61);

a data store associated with said device configured to store said transferred software in at least a portion of said data store (see Fig 3, Fig 4, steps 402, column 5, lines 55-56); and

a processor programmed to:

load said selected software to said device in response to said restart of said device (see Fig 4, steps 406, 414, column 6, lines 25-28, column 7, lines 4-9); and automatically revert, from said selected software to a different software version responsive to at least one of said selected software encountering a particular error (see column 7, lines 10-43).

Kraml does not explicitly teach that the computing device is a software defined radio device and the software and instructions are transferred via radio frequency (RF) communication. Kraml also does not teach the software server comprises a man-machine interface configured to receive instructions from a system operator.

Aghera is cited to teach updating software in a software defined radio device and the software and update instructions are transferred via radio frequency (RF) communication (see abstract, Fig 1, [0002], [0022]-[0024], [0053-[0060]), and a software server comprising a man-machine interface configured to receive instructions from a system operator (see [0026]).

It would have been obvious to one of ordinary skill in the art at the time of the invention to use the method of updating software as taught by Kraml in a software defined radio device using radio frequency (RF) communication as described by Aghera because Kraml does not limit the type of computing device on which the method of software updating can be performed (see column 3, lines 20-24, column 4, lines 36-40 of Kraml) and it is advantageous to use the updating method of Kraml in other types of computing devices such as a software defined radio device to benefit from the rollback feature such that if the new version of the software application is or becomes unusable for any reason, the device can quickly roll-back to the older version (see column 3, lines 33-37 of Kraml). In addition, it would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml such that the software server comprises a man-machine interface configured to receive instructions from a system operator as taught by Aghera to allow an operator to manage the update operations (see [0026] of Aghera).

Kraml and Aghera do not explicitly teach wherein said RF communication interface is further configured to receive a back-up copy of said transferred software that is executed in response to runtime errors, wherein said runtime errors are generated by executing said transferred software.

Cheston teaches a method of transferring software to a computer for execution, transferring, via network communication, a backup copy of the transferred software that is executed in response to runtime errors, wherein said runtime errors are generated by executing said transferred software (i.e., obtaining another copy of the application program after the application has become corrupted and crashed, see column 1, lines 34-51).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml and Aghera to transfer, via said RF communication, a back-up copy of said transferred software that is executed in response to runtime errors, wherein said runtime errors are generated by executing said transferred software as similarly taught by Cheston because executable applications often becomes corrupted and crashes during operation and another copy of the application is needed in order recover from the crash (see column 1, lines 38-51 of Cheston).

Kraml, Aghera, and Cheston do not teach uninstalling said selected software application and reverting back to said different version of said selected software application in response to a second fault.

Linam teaches uninstalling an update to a software component in response to an error detected during functional verification testing (see Fig 3, column 1, line 60 – column 2, line 7, column 3, lines 43-57, column 4, lines 10-43).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml, Aghera, and Cheston to uninstall said selected software application and reverting back to said different version of said selected software application in response to a second fault as taught by Linam such that the selected software application can be uninstalled based on functionality verification testing which would indicate software errors (see Fig 3, column 3, lines 43- column 4, line 42 of Linam) instead of waiting for the software to crash to revert to another version as taught by Kraml (see column 7, lines 10-29 of Kraml).

Kraml, Aghera, Cheston, and Linam do not explicitly teach that the reverting is performed without an instruction from said software server.

Aija teaches automatically reverting from a version of the software to a different software version without an instruction from a software server (see column 5, lines 51-60).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml, Aghera, Cheston, and Linam to automatically revert to a different software version without an instruction from said software server as taught by Aija such that the client device can initiate the recovery process after a system crash (see column 5, lines 51-53 of Aija).

28. As per claim 28, Kraml teaches wherein said processor is further programmed to determine that said software and said instruction are received successfully and to determine that said selected software is loaded successfully (see column 6, lines 29-46, column 7, lines 10-16).

29. As per claim 31, Kraml does not explicitly teach transferring software that comprises a plurality of software components.

Aghera teaches transferring software that comprises a plurality of software components (see [0032]).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml such that the transferred software comprises a plurality of software components as taught by Aghera because it is well known in the art to structure software as one or more components to conform to the well known practice of abstraction and encapsulation.

30. As per claim 32, Kraml does not explicitly wherein said RF communications interface is further configured to transmit a version identifying said software configured to be currently running in said software-defined radio device to said software server.

Aghera teaches transmitting a version identifying said software configured to be currently running in said software-defined radio device to said software server (see [0026], [0045]).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml to transmit a version identifying said software configured to be currently running in said software-defined radio device to said software server as taught by Aghera such that the patch server application can check if the wireless device needs a software patch (see [0045] of Aghera).

31. As per claim 33, Kraml does not explicitly teach receiving a software listing from said software-defined radio device, said software listing identifying software currently available on said data store.

Aghera teaches receiving a software listing from said software-defined radio device, said software listing identifying software currently available on said data store (i.e., patch profile maintains version information of all upgradeable software component on the wireless device, see [0030]).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Kraml to receive a software listing from said software-defined radio device, said software listing identifying software currently available on said data store as taught by Aghera such that the patch server will be able to determine whether or not the wireless device requires a certain patch (see [0030] of Aghera).

32. As per claim 34, Kraml as modified teaches a second data store associated with said software-defined device configured to store said transferred software (see column 5, lines 55-56).

33. As per claim 35, Kraml teaches wherein said second data store is nonvolatile (see column 5, lines 1-8).

34. As per claim 36, Kraml as modified teaches wherein said processor is further programmed to receive said software from said software server while said software-defined radio device performs software-defined radio functions (see column 65, line 65 - column 6, line 4).

35. Claim 25 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kraml et al. (US 6,141,683, hereinafter Kraml), in view of Aghera et al. (US 2004/0098715 A1, hereinafter Aghera), further in view of Cheston et al. (US 6,195,695 B1, hereinafter Cheston), further in view of Linam et al. (US 6,401,218 B1, hereinafter Linam), further in view of Simionescu et al. (US 2003/0084337 A1, hereinafter Simionescu).

36. As per claim 25, Kraml, Aghera, Cheston, and Linam do not explicitly teach decompressing the software after receiving the software.

Simionescu teaches decompressing a software at the host machine after receiving the software (see [0066]).

It would have been obvious to one of ordinary skill in the art at the time the invention to have modified Kraml, Aghera, Cheston, and Linam to compress and decompress the transferred software as taught by Simionescu because compression is a well known technique in the art to reduce the size of the software being transferred to reduce download time and bandwidth.

37. Claim 30 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kraml et al. (US 6,141,683, hereinafter Kraml), in view of Aghera et al. (US 2004/0098715 A1, hereinafter Aghera), further in view of Cheston et al. (US 6,195,695 B1, hereinafter Cheston), further in view of Linam et al. (US 6,401,218 B1, hereinafter Linam), further in view of Aija et al. (US 6,928,579 B2, hereinafter Aija), further in view of Simionescu et al. (US 2003/0084337 A1, hereinafter Simionescu).

38. As per claim 30, Kraml, Aghera, Cheston, Linam, and Aija do not explicitly teach a compression application for compressing the software prior to said software being transferred.

Simionescu teaches compressing a software prior to the software being transferred (see [0066]).

It would have been obvious to one of ordinary skill in the art at the time the invention to have modified Kraml, Aghera, Cheston, Linam, and Aija to compress and decompress the transferred software as taught by Simionescu because compression is a well known technique in the art to reduce the size of the software being transferred to reduce download time and bandwidth.

## ***Response to Arguments***

39. <u>Rejection of claims under §103(a)</u>:

40. As per independent claims 1, 16, 27, and 39, Applicants arguments have been fully considered but are moot in light of the new grounds of rejection.

## *Conclusion*

41. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Cheng et al. (US 6,151,643) is cited to teach uninstalling an update based on user dissatisfaction.
- Raghavan et al. (US 6,931,522 B1) is cited to teach a method for a computer using the system image on one of the partitions to boot itself into a known state in the event of a failure.
- Barfield et al. (US 6,948,166 B2) is cited to teach a method for automatically de-installing previously installed software based on user defined preferences.

42. Applicant's amendment necessitated the new ground(s) of rejection presented in this office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP §706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

43. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jue S. Wang whose telephone number is (571) 270-1655. The examiner can normally be reached on M-Th 7:30 am - 5:00pm (EST).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on 571-272-3759. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Tuan A Vu/
Primary Examiner, Art Unit 2193
11-30-2010

Jue Wang
Examiner
Art Unit 2193